***Картотека***

Написать программу - телефонную книжку. Каждая запись с информацией об абоненте должна содержать следующий набор полей:

|  |  |  |
| --- | --- | --- |
| Поле | Имя поля в структуре | Тип / Размер |
| ФИО | Name | char[50] |
| Телефон | Phone | int |
| Возраст | Age | int |
| Раздел | Group | enum (4 байта по умолчанию) |
| Дополнительные сведения | Misc | Индивидуально, в зависимости от раздела (реализовать в виде union) |

Поле «Раздел» может принимать одно из 3 значений, перечисленных с помощью enum:

GROUP\_FRIENDS

GROUP\_COLLEAGUES

GROUP\_APPLICANTS

Поле «Дополнительные сведения» зависит от раздела и должно быть реализовать в виде union:

|  |  |  |
| --- | --- | --- |
| Поле | Имя поля в структуре | Тип / Размер |
| **Друзья (Friends)** | | |
| День Рождения | Birthday | char[32] |
| Домашний адрес | Address | char[127] |
| Комментарии | Comments | char[255] |
| **Коллеги (Colleagues)** | | |
| Отдел | Department | enum (4 байта по умолчанию) |
| Номер рабочего места | CubicNo | int |
| Проект | Project | char[255] |
| **Соискатели (Applicants)** | | |
| Дата подачи заявления | ApllicationDate | char[32] |
| Язык программирования | Language | char[127] |
| Уровень знаний | Grade | int |
| Требуемая зарплата | Salary | int |

Поле «Отдел» может принимать одно из 4 значений, перечисленных с помощью enum:

DEPARTMENT\_SOFTWARE

DEPARTMENT\_HARDWARE

DEPARTMENT\_FIRMWARE

DEPARTMENT\_MANAGEMENT

## Требования к программе

Программа должна:

* выводить весь список на экран.
* предлагать простейшее меню действий в виде:
  1. Добавить
  2. Удалить
  3. Записать
  4. Считать
* **в режиме добавления** программа должна запросить последовательно все поля, причём в зависимости от введённого номера отдела (раздела) необходимо запрашивать соответствующие специфичные поля. В конце ввода информации она должна запросить номер записи, перед которой нужно добавить новую запись (0 – в самое начало; номер записи – после записи с этим номером; номер, превышающий количество записей – в самый конец картотеки).
* **в режиме удаления** программа должна запросить номер удаляемой записи, после чего удалить запись с таким номером.
* **в режиме записи** программа записывает всю картотеку в файл, предварительно запросив имя файла.
* **в режиме чтения** программа считывает всю картотеку из файла, предварительно запросив имя файла.

## Методические указания

Программа должна быть полностью оформлена в соответствии с принципами структурного программирования: программа должна быть разбита на функции, каждая из которых выполняет свою логически законченную задачу (не следует писать весь код полностью в функции main()). Для функций и логически завершённых участков кода необходимо расставить комментарии.

Для хранения данных **обязательно** используйте связанный список. Память для нового элемента должна выделяться и очищаться динамически. При загрузке списка из файла он должен загружаться в память **полностью**, и только после этого с ним разрешается работать (выводить на экран и т.п.)

Все перечисляемые параметры (например, тип отдела и тип транспорта), должны быть оформлены с помощью перечислений (enum).

Для описания информации, хранимой в одном элементе картотеки или записной книжки, необходимо создать структуру, включающую в себя как общие поля, так и наборы специфичных полей, объединённых с помощью union. Также в эту структуру должен входить указатель на следующий элемент для организации связанного списка.

Запись в файл и чтение из файла можно производить с помощью функций поблочного чтения и записи. В функцию блочной записи следует просто передавать адрес структуры и её размер, взятый с помощью оператора sizeof(). Однако следует помнить, что после чтения из файла указатель на следующий элемент связанного списка необходимо обновлять, так как., после чтения из файла расположение элементов связанного списка в памяти будет иным, нежели на момент записи элементов в файл. Поэтому адреса, хранящиеся в указателях на следующий элемент, должны будут отличаться от значений, хранящихся в них до записи.

Данные должны выводиться на экран в форматированном виде в виде таблицы, выделяя по 10 позиций для каждого поля. Слишком длинные значения должны обрезаться. Длина одной строки равна 80 символам. Поля, заданные с помощью перечислений, должны выводиться на экран не в числовом виде, а в виде соответствующих строковых значений (см. пример). В самом начале должен выводиться заголовок таблицы. Разделение колонок производите комбинацией символов “ | “.

Пример:

**# | Имя | Фамилия | Возраст | Отдел П || Проект | CPU | Memory |**

**| | | | М || Проект | Заказчик | Прогр-сты |**

**| | | | Д || Адрес | Заказчик | Вес | Транспорт**

**----------------+------------+------------+------------++------------+------------+------------+----------**

**01 | Андрей | Смирнов | 26 | Прогр. || TimeWarp | Athlon 2.4 | 512 |**

**02 | Алексей | Гагарин | 26 | Менеджмент || TimeWarp | Evil, Inc. | 3 |**

**03 | Андрей | Комягин | 26 | Доставка || Москва, 3 | Evil, Inc. | 24 | Авто**

Ниже приведён этот же пример в увеличенном виде:

# | Имя | Фамилия | Возраст | Отдел П || Проект

| | | | М || Проект

| | | | Д || Адрес

----------------+------------+------------+------------++---------01 | Андрей | Смирнов | 26 | Прогр. || TimeWarp 02 | Алексей | Гагарин | 26 | Менеджмент || TimeWarp 03 | Андрей | Комягин | 26 | Доставка || Москва,3

| CPU | Memory |

| Заказчик | Прогр-сты |

| Заказчик | Вес | Транспорт

+------------+------------+----------

| Athlon 2.4 | 512 |

| Evil, Inc. | 3 |

| Evil, Inc. | 24 | Авто